****"CHRDIS"****
BY MIKE SKALA

I'VE SEEN QUITE A BIT OF SOFTWARE LATELY UTILIZING THE "GRAPHIC CHARACTER MAKER", A MACHINE CODE ROUTINE THAT "ARCADIAN" HAS PUBLISHED IN THE PAST YEAR. THIS ALLOWED US TO USE A DISPLAY ROUTINE FROM THE ON-BOARD ROM AND PUT COMPLEX GRAPHICS ON THE SCREEN INSTANTLY. RATHER THAN A SLOW SERIES OF BOX AND LINE COMMANDS. THE MAJOR DRAWBACK HERE WAS WHEN MOVING THE GRAPHICS, ERASING AND REDRAWING LEFT US WITH CONSIDERABLE FLASHING OF BLINKING. IF YOU HAVE BEEN WITH US FOR A WHILE, YOU KNOW THAT WE ARE CONTINUALLY EVOLVING AND IMPROVING. THE FOLLOWING TUTORIAL IS OUR NEW GENERATION OF SCREEN ANIMATION FOR THE ASTROCADE!!!

ON-BOARD SUBROUTINE #51, "CHRDIS", IS A SINGLE CHARACTER DISPLAY ROUTINE MUCH AKIN TO THE ROUTINE USED IN THE GRAPHIC CHARACTER MAKER. WE CALL THE ROUTINE AS FOLLOWS:

```
DEC   HEX
255   FF   SYSSUK
51    33   CHRDIS
N     N    E-HOR POS. (0-159 DEC.)
N     N    D-VER POS. (0-99 DEC.)
N     N    C-CHAR DISPLAY PARAMETER
N     N    A-CHAR TO CALL
```

SYSSUK/CHRDIS--ANY ON-BOARD SUBROUTINE CAN BE CALLED IN ONE OF TWO WAYS: "SYSTEM" ASSUMES ALL NECESSARY INFO IS ALREADY IN THE REGISTERS. (MEMORY CELLS WITHIN THE C.P.U.) "SYSSUK" MEANS THE INFO WILL BE DIRECTLY FOLLOWING THE CALL, AND BE "SUCKED" IN.

E--THIS REGISTER IS THE HORIZONTAL COORDINATE FOR YOUR GRAPHIC. ZERO IS THE LEFT SIDE OF THE SCREEN, AND 159 IS THE RIGHT SIDE. IF YOU GO PAST 159, THE GRAPHIC WILL REAPPEAR ON THE LEFT SIDE, ONE PIXEL LOWER. THIS IS FINE UNTIL YOU REACH 256, ABOUT HALF WAY ACROSS THE SCREEN, WHERE IT WILL AGAIN DROP ONE PIXEL DOWN AND BACK OVER TO THE LEFT SIDE OF THE SCREEN. FOR THIS REASON, IT IS BEST TO LIMIT-CHECK THE GRAPHIC TO BETWEEN 0 AND 159.

D--THIS REGISTER IS THE VERTICAL COORDINATE. ZERO IS THE TOP OF THE SCREEN, 99 THE BOTTOM. BE CAREFUL NOT TO RUN YOUR GRAPHIC DOWN INTO THE SCRATCHPAD AREA HIDDEN AT THE BOTTOM OF THE SCREEN. LIMIT-CHECK YOUR GRAPHIC AGAIN NOT TO RUN OFF THE BOTTOM, OR THE WHOLE PROGRAM MIGHT CRASH!! HERE'S SOMETHING INTERESTING: THERE IS A WHOLE OTHER SCREEN ABOVE THE ONE WE NORMALLY SEE. YOU'LL NEVER SEE IT, BUT YOU CAN MOVE THINGS AROUND UP THERE WITH NEGATIVE VALUES IN THIS REGISTER. YOU COULD, FOR EXAMPLE, START A GRAPHIC UP THERE AND HAVE IT FALL DOWN INTO THE VISIBLE SCREEN.

C--THIS CONTROLS A LOT OF INTERESTING THINGS, AND WE'LL GO INTO DETAIL IN A FUTURE ARTICLE. FOR NOW USE 40 (28 HEX), WHICH GIVES US AN XOR WRITE.

A--THIS IS WHICH CHARACTER WE ARE GOING TO DISPLAY. IT RESPONDS TO THE STANDARD ASCII CODE TO DISPLAY ALL RESIDENT CHARACTERS, 0 THROUGH 127, OR OUR OWN THAT WE CAN CREATE.

NOW THAT YOU UNDERSTAND ALL THAT STUFF, LET'S ASSEMBLE A SMALL MACHINE CODE PROGRAM AT THE BACK END OF OUR LINE INPUT BUFFER. THE FIRST THING YOU MUST DO WHEN GOING INTO A MACHINE CODE PROGRAM FROM BASIC IS TO SAVE THE DE REGISTER. THIS IS A LITTLE MEMORY CELL THAT REMEMBERS WHERE YOU WERE BEFORE YOU LEFT. NOW WE GO INTO OUR CHRDIS ROUTINE. NOW COMES THE NEW TWIST! WHEN WE RETURN INTO ANOTHER CHRDIS, BRING BACK OUR DE REGISTER, AND RETURN TO BASIC. THEREFORE, IF WE ALREADY HAVE OUR GRAPHIC ON THE SCREEN, THIS WILL ERASE THE OLD AND REDRAW THE NEW IN ONE CALL, YIELDING MINIMUM "OFF" TIME.

ENTER THE FOLLOWING DIRECT COMMAND (NO LINE NUMBER), AND THEN INPUT THE DECIMAL VALUES LISTED BELOW:

```
NT=1:FOR A=26241TO 26257:PRINT A,:INPUT
" ",B:%(A)=B:PRINT :NEXT A
```

```
              DEC   HEX
%(26241)=213   D5   PUSH DE
%(26242)=255   FF   SYSSUK
%(26243)= 51   33   CHRDIS
%(26244)=  0    0   E (HOR.)
%(26245)=  0    0   D (VER.)
%(26246)= 40   28   C
%(26247)=  0    0   A (CHAR#)
%(26248)=  0    0   NOP (NO OPERATION)
%(26249)=255   FF   SYSSUK
%(26250)= 51   33   CHRDIS
%(26251)=  0    0   E (HOR.)
%(26252)=  0    0   D (VER.)
%(26253)= 40   28   C
%(26254)=  0    0   A (CHAR#)
%(26255)=  0    0   NOP
%(26256)=209   D1   POP DE
%(29257)=201   C9   RET
```

NOW WE NEED A LITTLE BASIC PROGRAM TO MANIPULATE ALL THIS STUFF. ENTER THE FOLLOWING:

```
10 %(26244)=9999:V=0:H=0
20 V=V-JY(1):H=H+JX(1)
30 IF V<0V=0
40 IF V>75V=75
50 IF H<0H=0
60 IF H>159H=159
70 %(26252)=V%256+H
80 C=%(28)*65+54:%(26254)=C
90 CALL26241:%(26244)=%(26251):%(26247)=C:
GOTO 20
```

LINE 10 WE SET THE COORDINATES FOR THE FIRST
CHRDIS UPSTAIRS SOMEWHERE OUT OF SIGHT, AND
THEN THE SECOND CHRDIS WILL DRAW THE INITIAL
GRAPHIC. THIS LINE IS ONLY RUN ONCE, AFTER
WHICH THE PROGRAM WILL USE THE FIRST CHRDIS
TO ERASE THE OLD GRAPHIC. LETTER VARIABLES V
AND H ARE ALSO ZEROED OUT HERE.

LINE 20 PROGRAM READS THE JOYSTICKS TO UP-
DATE VARIABLES USED FOR VERTICAL AND HORIZON-
ITAL COORDINATES.

LINES 30-60 LIMIT CHECKS THE GRAPHIC TO KEEP
IT ON THE SCREEN

LINE 70 HERE IS WHERE THE NEW COORDINATES
ARE PLUGGED INTO OUR SECOND CHRDIS. SINCE ANY
POKE WORKS ON TWO MEMORY LOCATIONS, WE MUST
USE THE FORMAT VD256+H.

LINE 90 FN(1) IS READ, AND VARIABLE C WILL
BE SET TO 94,95,96 OR 97. THESE ARE THE ASCII
VALUES FOR THE FOUR ARROWS FOUND ON OUR KEY-
PAD. THIS VALUE IS THEN POKED INTO THE CHAR#
POSITION OF OUR SECOND CHRDIS. IN CASE YOU'VE
WONDERED, WHY A "NOP" FOLLOWED THE A REGISTER
IN BOTH CHRDIS'S, IT'S AGAIN BECAUSE OF OUR
POKE SITUATION. A NOP IS A MACHINE CODE COM-
MAND THAT DOES ABSOLUTELY NOTHING EXCEPT
WASTE A BYTE WERE IT NOT THERE IN THIS PRO-
GRAM, WE SIMPLY COULD NOT POKE IN OUR ASCII
VALUE. WE WOULD INSTEAD END UP WITH SOME
LARGE NEGATIVE NUMBER TO POKE IN, AND THE
FIRST AND SECOND CHRDIS'S WOULD REQUIRE DIFF-
ERENT FUDGE FACTORS.

LINE 90 THE MACHINE CODE PROGRAM IS NOW
CALLED. THE OLD COORDINATES ARE SET EQUAL TO
THE NEW ONES, THE OLD CHAR# IS SET EQUAL TO
THE NEW ONE, AND THE PROGRAM LOOPS BACK TO
LINE 20.

I WOULD SUGGEST THAT YOU DUMP THE PROGRAM
TO TAPE BEFORE YOU RUN IT. BECAUSE ONE LITTLE
MISTAKE WITH MACHINE CODE CAN CAUSE BIG PRO-
BLEMS TO DUMP. USE THE FOLLOWING COMMAND:
PRINT ; PRINT %(26241),17
TO LOAD IT BACK FROM TAPE, USE:
INPUT ; INPUT %(26241)
RUN THE PROGRAM AND FIDDLE WITH THE JOY-
STICK AND KNOB. I'LL ADMIT IT ISN'T A GAME OR
LOTS OF FUN. BUT IT DEMONSTRATES A FAIRLY
SIMPLE MEANS OF SMOOTH ANIMATION. THIS WILL
WORK WITH HOMEMADE CHARACTERS AS WE DID IN
THE GRAPHIC CHARACTER MAKER, AND THIS WILL
BE COVERED IN THE NEXT TUTORIAL.

ONE MAJOR PRECAUTION MUST BE OBSERVED WITH
THIS ROUTINE. WHEN USING THE NEW ASTRO BASIC,
THE PROGRAM ACTIVELY USES THE 104 BYTE LINE
INPUT BUFFER. SINCE WE HAVE STORED OUR MACH-
INE CODE IN THE LAST 17 BYTES OF THE BUFFER,
NO LINE IN OUR BASIC PROGRAM CAN BE MORE THAN
87 BYTES LONG!!!!!

YOU WILL NOTICE A SLIGHT FLICKER OF OUR
GRAPHIC. THIS HAS SOMETHING TO DO WITH TIMING
OR SCREEN INTERRUPTS OR SOMETHING. I DON'T
KNOW. I'M SURE SOMEONE OUT THERE KNOWS THE
CURE, SO PLEASE WRITE IN!!

VARIATIONS:
"WRAPAROUND"
30 IF V<0V=75
40 IF V>75V=0
50 IF H<0H=159
60 IF H>159H=0

"BIG TIME"
5 %(20245)=163:%(20253)=163
40 IF V>65V=65

"LETTER DROP"
86 C=&(28)C3+33;%(20254)=C
90 CALL20241;%(20244)=X:%(20251):;%(20247)=C; IF
TRC()GOTO 10
100 GOTO 20

MIKE SKALA
544 E. OVERLOOK
EASTLAKE, OH
44094

ARCADIAN

CHRDIS II
HOW TO USE HOME MADE GRAPHICS
BY MIKE SKALA

BUILDING YOUR OWN GRAPHICS FOR USE WITH OUR "CHRDIS" ROUTINE IS A FAIRLY SIMPLE TASK. THE "GRAPHIC CHARACTER MAKER" PUBLISHED IN THE ARCADIAN (VOL.3 PP.82-84) COULD BE MODIFIED READILY IF YOU KNOW WHAT YOU ARE DOING. IF YOU DON'T, THEN READ ON...

THE "CHRDIS" WILL LOOK AT YOUR CHARACTER IN BLOCKS ONE PIXEL HIGH BY EIGHT PIXELS WIDE, WITH EACH PIXEL BEING EITHER "OFF" (FC) OR "ON" (FC). YOU MUST FIGURE OUT THE VALUE OF EACH BLOCK BY TOTALLING THE "PIXEL VALUES".

128	64	32	16	8	4	2	1

FIG. 1

(REFER TO FIG.1). IF A PIXEL IS "ON", IT'S VALUE IS ADDED TO THE TOTAL. FOR EXAMPLE, ALL EIGHT PIXELS "ON" WOULD HAVE A BLOCK VALUE OF 255, ALL "OFF", A VALUE OF ZERO, OR JUST THE FOUR ON THE RIGHT HAND SIDE "ON" WOULD EQUAL 15.(8+4+2+1=15) LET'S CREATE A SMALL GRAPHIC TO ILLUSTRATE. LOOK AT FIG.#2 TO SEE HOW WE GOT OUR BLOCK VALUES. YOU CAN GO EITHER HIGHER, WIDER, OR BOTH, AND BLOCK VALUES WILL BE READ FROM LEFT TO RIGHT,(IF MORE THAN ONE BLOCK WIDE), AND TOP TO BOTTOM.
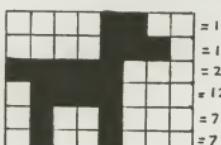
= 12
= 14
= 248
= 120
= 72
= 72

FIG. 2

SO NOW THAT YOU HAVE SOME BLOCK VALUES, WHERE DO THEY GO? WELL, WITHOUT SPECIAL MEMORY, (E.G. A BLUE RAM, ETC.), YOU HAVE TWO CHOICES. EITHER WE STORE THEM IN THE LINE INPUT BUFFER WITH THE REST OF OUR MACHINE CODE, OR AT THE VERY BOTTOM OF OUR SCREEN. THE FORMER GETS RATHER CROWDED IN A HURRY, OR THE LATTER CAN GET WIPED OUT BY CLEARING THE SCREEN OR RUNNING GRAPHICS INTO THE BOTTOM. DIRECTLY FOLLOWING THE LINE INPUT BUFFER IS A MEMORY AREA CALLED THE "STACK". THIS IS SORT OF A "PARKING LOT" FOR BASIC TO STORE AND RETRIEVE DATA. SELDOM DOES THIS AREA GET FILLED UP, SO WE CAN GENERALLY RUN A FEW DOZEN BYTES INTO THIS AREA WITHOUT PROBLEMS. THE BEST APPROACH HERE IS TO PUT OUR GRAPHIC INFO IN THE DEEP END, AND OUR MACHINE CODE ROUTINE UP IN THE SAFE END. THIS WAY, IF THE STACK RUNS OVER OUR GRAPHICS, WE GET FUNNY LOOKING CHARACTERS, WHEREAS RUNNING OVER OUR MACHINE CODE ROUTINE WOULD CAUSE OUR PROGRAM TO BOMB.

WHAT WE ARE DOING HERE IS CREATING AN ALTERNATE CHARACTER FONT WITH THE MUTT BEING OUR FIRST AND ONLY CHARACTER. WE START OUR LIST OF CHARACTERS WHERE ASCII CODES END, SO HE WILL BE CHARACTER NUMBER #128. TO USE THIS FONT, WE NOW HAVE A NEW RESPONSIBILITY. WE MUST CONSTRUCT A TABLE IN MACHINE CODE THAT TELLS OUR COMPUTER ALL ABOUT THIS NEW FONT, AND LET IT KNOW WHERE WE HID THIS TABLE. LOAD THE DECIMAL VALUES BELOW WITH THIS DIRECT COMMAND:

```
FOR A=20237TO 20270:CY=0;PRINT A,;
INPUT " ",%(A);BOX 0,0,160,20,2;
NEXT A
```

	DEC	HEX	
%(20237)=221	DD	⎫ LD IX,nn	LOAD IX WITH ADDR
%(20238)= 33	21	⎬	OF OUR ALTERNATE
%(20239)= 34	22	n ⎭	FONT TABLE.
%(20240)= 79	4F	n	(20258)
%(20241)=213	D5	PUSH DE	
%(20242)=255	FF	SYSSUK	
%(20243)= 51	33	CHRDIS	
%(20244)= 00	00	E (HOR)	
%(20245)= 00	00	D (VER)	
%(20246)= 40	28	C	
%(20247)= 00	00	A (CHAR#)	SAME AS
%(20248)= 00	00	NOP	"CHRDIS"
%(20249)=255	FF	SYSSUK	
%(20250)= 51	33	CHRDIS	
%(20251)= 00	00	E (HOR)	
%(20252)= 00	00	D (VER)	
%(20253)= 40	28	C	
%(20254)= 00	00	A (CHAR#)	
%(20255)= 00	00	NOP	
%(20256)=209	D1	POP DE	
%(20257)=201	C9	RET	
%(20258)=128	80	# OF OUR ALT. CHARACTER	
%(20259)= 00	00	(WE DON'T USE 'EM)	
%(20260)= 00	00	CHR SIZE WIDTH (K9 PIXELS-1 BLOCK)	
%(20261)= 1	01	CHR SIZE WIDTH	
%(20262)= 6	06	CHR SIZE HEIGHT	
%(20263)= 41	29	⎫ STARTING ADDR. OF	
%(20264)= 79	4F	⎬ BLOCK VALUES (20265)	
%(20265)= 12	0C	⎭	
%(20266)= 14	0E		
%(20267)=248	F8	⎫ BLOCK VALUES	
%(20268)=120	78	⎬ FOR OUR PUP.	
%(20269)= 72	48		
%(20270)= 72	48		

NOW ENTER THE FOLLOWING BASIC PROGRAM:

```
>10 %(20244)=-9999;V=0;H=0;C=128;%(20247)=C;
%(20254)=C
>20 V=V-JY(1);H=H+JX(1)
>30 IF V<0V=0
>40 IF V>82V=82
>50 IF H<0H=0
>60 IF H>152H=152
>70 %(20251)=V%256+H
>80 CALL20237
>90 %(20244)=%(20251)
>100 GOTO 20
```

37

THE FOLLOWING PARAGRAPH & MODIFICATIONS
WERE ACCIDENTALLY LEFT OUT OF "CHRDIS II" IN
LAST MONTH'S ISSUE:

NOTICE THAT OUR CALL IS MADE NOW TO
%(20237) SO AS TO LOAD THE IX REGISTER WITH
OUR ALT. FONT TABLE ADDRESS. SINCE YOU ARE
GOING TO WANT BIGGER AND/OR MORE CHARACTERS,
YOU MAY HAVE TO GO DEEPER INTO THE STACK
AREA, OR EXPERIMENT WITH STORING STUFF AT
THE BOTTOM OF THE SCREEN. JUST BE SURE TO
LOAD THE ALT. FONT TABLE WITH THE PROPER CHR
HEIGHT & WIDTH AND THE CORRECT ADDRESS OF
YOUR BLOCK VALUES.

MAKE YOUR HOUND HAPPY WITH THE FOLLOWING
MODIFICATIONS:

```
%(20271)=12    ADD LINES: >25 C=C+1; IF C>129
%(20272)=142                  C=128
%(20273)=120              >26 %(20254)=C
%(20274)=120              >95 %(20247)=
%(20275)=72                   %(20254)
%(20276)=72
```

* *

CHRDIS III
BY MIKE SKALA
CHARACTER DISPLAY PARAMETERS

THIS IS THE THIRD AND FINAL SEGMENT IN
MY THREE PART SERIES EXPLAINING THE ON-BOARD
SUBROUTINE "CHRDIS". WE WILL FINALLY LOOK AT
THE "C" BYTE WITHIN "CHRDIS", WHICH IS AN
OPTION BYTE USED TO CONTROL THREE SEPERATE
FUNCTIONS IN ALL SYSTEM ALPHANUMERIC DISPLAY
ROUTINES; SIZE, TYPE OF SCREEN WRITE, AND
COLOR.

FIRST WE BETTER BE SURE WE UNDERSTAND
HOW IMAGES AND OUR BASIC TEXT ARE USING OUR
RAM TO ENSURE WE DON'T LET THEM CONFLICT.
OUR TEXT IS STORED IN THE EVEN NUMBER BITS
OR BITS. THAT MEANS WE CAN SET AN ODD BIT TO
ONE, OR TURN IT ON (FC), OR WE CAN RESET
THE BIT TO ZERO, OR TURN IT OFF (BC). SINCE
THESE ONBOARD SUBROUTINES WERE DESIGNED FOR
FOUR IMAGES, THEY ARE ALWAYS WORKING
WITH TWO BITS AT A TIME, I.E., THEY ARE
GOING TO LOOK AT THE GRAPHICS THAT WERE
BUILT IN A ONE BIT PER PIXEL FORMAT AND EX-
PAND THEM INTO TWO BITS PER PIXEL. FIG. #1
SHOWS A TYPICAL BYTE OF RAM, AND FIG. #2

FIG.
#1

G	T	G	T	G	T	G	T

BIT#   7   6   5   4   3   2   1   0

SHOWS THE FOUR POSSIBLE COMBINATIONS WE CAN
EXPAND INTO. WE HAVE THREE DIFFERENT WAYS WE
CAN DO A WRITE: 'PLOP', 'OR', AND 'XOR'.
'PLOP' (FIG.#3) WILL SIMPLY REPLACE OUR RAM
VALUE WITH OUR IMAGE VALUE. YOU CAN SEE THAT
THIS WILL DESTROY OUR TEXT BITS, SO FORGET
ABOUT 'PLOPPING'. AN 'OR'(FIG.#4) WILL FIRST
LOOK AT THE RAM, AND THE RESULT WILL BE 'ON'
IF EITHER OR BOTH THE IMAGE BIT AND RAM BIT
WAS ON. IF BOTH WERE OFF, THE RESULT WILL BE
OFF. AN 'XOR' (FIG.#5 A&B) IS AN EXCLUSIVE
'OR'. THIS WILL SET THE RESULT IF THE IMAGE

PLOP
FIG. 3

	1	0
RAM	0	1
RESULT	0	1

OR
FIG. 4

	1	0
RAM	0	1
RESULT	1	1

XOR
FIG. 5

A	1	0	B	1	0
RAM	0	1	RAM	1	1
RESULT	1	1	RESULT	0	0

BIT OR THE RAM BIT WAS ON, BUT RESET IF BOTH
WERE ON. THIS IS WHAT WE'VE BEEN USING IN
THE PREVIOUS TUTORIALS, WHERE YOU 'XOR' AN
IMAGE TO A BLANK AREA AND THE GRAPHIC APP-
EARS, THEN 'XOR' IT TO THE SAME PLACE AND IT
IS ERASED. I HOPE YOU CAN SEE BY NOW THAT IF
WE EXPAND INTO 00 OR 10 AND USE 'OR' & 'XOR'
WE CAN STILL WRITE TO THE SCREEN WITHOUT
DISTURBING OUR TEXT. IF THIS IS STILL CON-
FUSING, JUST REMEMBER NOT TO EXPAND INTO 01
OR 11 AND DON'T 'PLOP'. YOU MAY ALSO WISH TO
LEARN MORE ABOUT BINARY LOGIC (OR, XOR,ETC.)
AS IT IS USED QUITE A BIT IN MACHINE CODE.
LET'S PROCEED. FIG. #6 SHOWS HOW THE 'C'
BYTE IS CONSTRUCTED. STARTING BACKWARDS (AS
ALWAYS) BITS 1 & 0 ARE WHAT AN OFF BIT IN
OUR GRAPHIC WILL BE EXPANDED INTO (00 HERE

FIG. 6

ENLARGE FACTOR	XOR	OR	ON COLOR	OFF COLOR

BIT#   7   6   5   4   3   2   1   0

WILL SET IT EQUAL TO BC). BITS 3 & 2 ARE
WHAT AN ON BIT IS EXPANDED INTO (10 WILL
EQUAL FC). YOU COULD SET BITS 3 & 2 TO 00,
AND BITS 1 & 0 TO 10 AND GET A "REVERSE"
IMAGE.

BITS 5 & 4 CONTROL THE TYPE OF SCREEN
WRITE. DON'T SET EITHER AND YOU'LL GET A
'PLOP' (A NO-NO), SET BIT 5 TO GET AN 'XOR',
SET BIT 4 TO GET AN 'OR'. I THINK WE'VE
KILLED THIS SUBJECT ALREADY.

BITS 7 & 6 CONTROL A CUTE TRICK WE'VE
SEEN AS FAR BACK AS VOL. #1 OF THE ARCADIAN,
AND IN NEARLY EVERY GAME CARTRIDGE. THIS
ALLOWS US TO DISPLAY ANY CHARACTER (ASCII OR
HOMEMADE) IN NORMAL SIZE OR IN AN ENLARGED
FASHION. (SEE FIG.#7). TWO THINGS TO REMEM-
BER HERE, THE ADDRESS OR LOCATION OF THE
CHARACTER ALWAYS REFERS TO THE UPPER LEFT
HAND CORNER, NOT THE CENTER, AND KEEP YOUR
IMAGE OUT OF THE SCRATCHPAD AREA AT THE BOT-
TOM OF THE SCREEN.

FIG. 7

BIT#		FINAL	SO JUST BUILD THIS "C"
7	6	SIZE	BYTE IN BINARY AND CON-
0	0	NORMAL	VERT TO DECIMAL. LIKE WE
0	1	2X	CONVERTED OUR BLOCK VAL-
1	0	4X	UES FOR OUR PUPPY IN THE
1	1	8X	LAST ARCADIAN.BE SURE TO

PUT THE PROPER "C" BYTE
IN BOTH THE "CHRDIS" IN
YOUR MACHINE CODE ROUTINE

WELL, THIS SHOULD GIVE YOU HACKERS SOME
HANDY TOOLS TO GENERATE SOME PRETTY CLASSY
PROGRAMMING. IF YOU HAVE ANY FURTHER QUES-
TIONS YOU CAN CONTACT ME DIRECT BY MAIL OR
PHONE (EVENINGS). ALSO, DON'T BE AFRAID TO
SHOW APPRECIATION BY SENDING A FEW OF YOUR
PROGRAMS. I'D LOVE TO SEE THEM!!!
    MIKE SKALA 544 E. OVERLOOK
    EASTLAKE, OHIO 44094 (216) 951-2564

                              © R.Fabris 1983